
CCCCCCCCCCC
CCCCCCCCCCC
CCCCCCCCCCC
CCC
CCC
CCC
CCC
CCC
CCC
CCC
CCC
CCC
CCC
CCC
CCC
CCC
CCC
CCC
CCCCCCCCCCC
CCCCCCCCCCC
CCCCCCCCCCC
JJJ
JJJ
JJJ
JJJ
JJJ
JJJ
JJJJJJJJJ
JJJJJJJJJ
JJJJJJJJJ
JJJ
JJJ
JJJ
JJJ
JJJ
JJJ
JJJ
JJJ
JJJ
FFFFFFFFFFFFFFF
FFFFFFFFFFFFFFF
FFFFFFFFFFFFFFF
FFF
FFF
FFF
FFF
FFF
FFF
FFFFFFFFFFFF
FFFFFFFFFFFF
FFFFFFFFFFFF
FFF
FFF
FFF
FFF
FFF
FFF
FFF
FFF
FFF
VVV VVV
VVV VVV
VVV VVV
VVV
VVV
VVV
VVV VVV
VVV VVV
VVV VVV
VVV
VVV
VVV
444 444
444 444
444 444
444 444
444 444
444 444
444 444
444444444444444
444444444444444
444444444444444
444
444
444
444
444
444
444
444
444

```
          JJ  NN       NN  LL          PPPPPPPP    RRRRRRRR    EEEEEEEEEE  FFFFFFFFFF    IIIIII    XX      XX
          JJ  NN       NN  LL          PPPPPPPP    RRRRRRRR    EEEEEEEEEE  FFFFFFFFFF    IIIIII    XX      XX
          JJ  NN       NN  LL          PP      PP  RR      RR  EE          FF              II      XX      XX
          JJ  NN       NN  LL          PP      PP  RR      RR  EE          FF              II      XX      XX
          JJ  NNNN     NN  LL          PP      PP  RR      RR  EE          FF              II        XX  XX
          JJ  NNNN     NN  LL          PP      PP  RR      RR  EE          FF              II        XX  XX
          JJ  NN  NN   NN  LL          PPPPPPPP    RRRRRRRR    EEEEEEEE    FFFFFFFF        II          XX
          JJ  NN  NN   NN  LL          PPPPPPPP    RRRRRRRR    EEEEEEEE    FFFFFFFF        II          XX
JJ        JJ  NN    NNNN  LL          PP          RR  RR      EE          FF              II        XX  XX
JJ        JJ  NN    NNNN  LL          PP          RR  RR      EE          FF              II        XX  XX
JJ        JJ  NN       NN  LL          PP          RR    RR    EE          FF              II      XX      XX      ....
JJ        JJ  NN       NN  LL          PP          RR    RR    EE          FF              II      XX      XX      ....
  JJJJJJ      NN       NN  LLLLLLLLLL  PP          RR      RR  EEEEEEEEEE  FF          IIIIII    XX      XX      ....
  JJJJJJ      NN       NN  LLLLLLLLLL  PP          RR      RR  EEEEEEEEEE  FF          IIIIII    XX      XX      ....
```

```
RRRRRRR       333333       222222
RRRRRRRR     333333       222222
RR      RR  33      33   22      22
RR      RR  33      33   22      22
RR      RR          33           22
RR      RR          33           22
RRRRRRRR          33           22
RRRRRRRR          33           22
RR   RR           33       22
RR   RR           33       22
RR    RR    33    33     22
RR    RR    33    33     22
RR      RR    333333    2222222222
RR      RR    333333    2222222222
```

```
! IDENT = "V04-000"
!*****************************************************************************
!*                                                                           *
!*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
!*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
!*  ALL RIGHTS RESERVED.                                                     *
!*                                                                           *
!*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
!*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE           *
!*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER      *
!*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
!*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY        *
!*  TRANSFERRED.                                                             *
!*                                                                           *
!*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE     *
!*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT         *
!*  CORPORATION.                                                             *
!*                                                                           *
!*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS      *
!*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
!*                                                                           *
!*                                                                           *
!*****************************************************************************

!++
! FACILITY: Common Journaling Facility (CJF)

! ABSTRACT:
!               LIBRARYs and REQUIREs


! ENVIRONMENT:

! AUTHOR: CJF group

! MODIFIED BY:

!       V03-031 EMD0005         Ellen Dusseault         26-SEP-1983
!               Set JNL$C_MAX_COPIES to 1.
!
!       V03-030 MKL0155         Mary Kay Lyons          25-Jul-1983
!               Delete JNL$_xxx message names. Delete V3 conditionals.
!
!       V03-029 JSV0338         Joost Verhofstad        28-JUN-1983
!               Require CJF$ message from .R32 file
!
!       V03-028 JSV0268         Joost Verhofstad        18-MAY-1983
!               Add CJF$_JNLNAMTLNG and CJF$_ACPNAMTLNG and
!               convert JNL$_ to CJF$_
!
!       V03-027 JSV0238         Joost Verhofstad        29-APR-1983
!               Add CJF$_JNLNOTGRP
```

```
        V03-026 JSV0209         Joost Verhofstad        06-APR-1983
                Change NEXT_STAGE so it can be called from loops

        V03-025 MKL0064         Mary Kay Lyons          30-MAR-1983
                Add declarations for CJF$_ and JNL$_: NOSUCHVER,
                NVERR, NEWPROL, OLDPROL, CPYNOTAVL, OLDVERSION.

        V03-024 LY0322          Larry Yetto             9-MAR-1983 15:10:45
                Fix spelling of CJF$_POSJNL.  Put P1 allocation macros back.

        V03-023 LY0316          Larry Yetto             8-MAR-1983 14:41:33
                Add binds equating JNL$_ symbols for all CJF$_
                messages.  At some time in the future the messages themselves
                will be CJF$ but by equating the symbols now we can slowly
                phase in the new symbols.

        V03-022 JSV0147         Joost Verhofstad        17-FEB-1983
                Add declaration of CJF$_INVTMPF and CJF$_BATJONLY
                and CJF$_INVITMLST

        V03-021LY0303           Larry Yetto             16-FEB-1983 11:09:29
                Back out P1 allocation until the exec routines are fixed

        V03-020 LY0296          Larry Yetto             06-Feb-1983
                Modify SERVICE_INIT_STAGE, SERVICE_END_STAGE and DEFINE_OFFSETS
                to allocate memory from P1 instead of using EXPREG.
                Add CJF$UNLOCK_PROTO macro

        V03-019 JSV0137         Joost Verhofstad        03-FEB-1983
                Replace source, put in null packet

        V03-018 JSV0117         Joost Verhofstad                05-Jan-1983
                Add CJF$_EXRUJQUOTA

        V03-017 LY0231          Larry Yetto                     09-Dec-1982
                Add CJF$_FILEXI error code declaration.  Modify
                SERVICE_INIT_STAGE and DEFINE_OFFSETS to ignore the
                first longword in the allocated memory

        V03-016 JSV0101         Joost Verhofstad                01-Dec-1982
                Add declarations for error codes to replace INVPAR
                in JNLACP

        V03-015 LY0218          Larry Yetto
                Require JNLDEF.R32 if building a V3.x system.  Modify
                NEXT_STAGE to put the stage data into a buffer area
                which is set up in one of the INIT_STAGE macros.  The
                area was changed from OWN to LOCAL to make the services
                reentrant, however, the BLISS compiler then started
                reusing stack locations for successive NEXT_STAGE macros
                which tended to cause strange occurances when we run down
                through the next stage routines.

        V03-014 JSV0092         Joost Verhofstad        04-Nov-1982
                Add CJF$_PREMEOF message declaration
```

```
        V03-013 LY0200          Larry Yetto             1-Nov-1982
                Add LAST_STAGE_NUMBER compile time value so that
                PREVIOUS=FINAL in NEXT_STAGE doesn't print bogus messages.
                Add CJF$_NOTIMFLTEL message number.

        V03-012 LY0172          Larry Yetto             22-Oct-1982
                Add external literal definitions for messages to replace
                INVPAR.

        V03-011 LY0135          Larry Yetto             20-Oct-1982
                Add CREDAT field to FILE_BLCK_FIELDS.  Remove OWN data
                from INIT_STAGE and NEXT_STAGE.  Add NAMTBL_BUFF_LEN and
                NAMTBL_BUFF_BLKS literal definitions.
                Modify code so that it is rentrant and can be loaded
                into system space as a as a system service.

        V03-010 LY0126          Larry Yetto             16-Sep-1982
                Remove references to STAGE$... global symbols in INIT_STAGE
                and NEXT_STAGE macros.  Add USERMODE_INIT_STAGE macro.

        V03-009 JAY0002         John A. Ywoskus         31-Aug-1982
                Conditionally require in the V3BLDREQ file to
                resolve symbols for V3.x builds.

        V03-008 LY0101          Larry Yetto             25-Aug-1982
                Remove CJF$_TRUNC message.

        V03-007 JSV0042         Joost Verhofstad        10-Aug-1982
                Add declaration for CJF$_RUCONTROL, CJF$_ZEROEXT

        V03-006 JSV0032         Joost Verhofstad        28-Jul-1982
                Remove temporary definitions

        V03-005 GJA0011         Greg Awdziewicz         27-Jul-1982  19:37
                Remove JNLDEF require declaration.

        V03-004 LY0050          Larry Yetto             27-Jul-1982
                Add return code external definitions.  Add file block
                field, literal, and structure definitions.

        V03-004 JSV0023         Joost Verhofstad        16-Jul-1982
                Add return codes to be declared

        V03-003 LY0041          Larry Yetto             12-Jul-1982
                Remove temporary definition for VCB$W_JNL_MXENT

        V03-002 JAY0001         John A. Ywoskus         08-Jul-1982
                Add ENTTOOBIG error.

        V03-001 LY0036          Larry Yetto  1-JUL-1982
                Change message definitions from requiring a .B32 file
                to defining them as external. Add the copywrite.  Add
                temporary definition for VCB$W_JNL_MXENT
;--
```

```
LIBRARY 'SYS$LIBRARY:LIB' ;

! REQUIRE 'SHRLIB$:CJFMSG' ;
REQUIRE 'SHRLIB$:BLIOPTS.R32' ;
REQUIRE 'SHRLIB$:PSECTS' ;
REQUIRE 'SHRLIB$:JNLDEFINT' ;
REQUIRE 'SHRLIB$:JNLFILE' ;
REQUIRE 'SHRLIB$:CJFMSG' ;
```

```
!++
! BUILTIN declarations
!--

BUILTIN
    CHMU,
    MTPR,
    INSQUE,
    REMQUE ;

!++
! Declarations of EXEC routines used in many places in Journaling
!--
    LINKAGE
        CVT_DEVNAM      = JSB (
            REGISTER = 0,                   ! length of output buffer
            REGISTER = 1,                   ! address of output buffer
            REGISTER = 4,                   ! value for format of name returned
            REGISTER = 5 ;                  ! address of device UCB
            REGISTER = 1 ) : NOPRESERVE ( 2 ) ;
    EXTERNAL ROUTINE
        IOC$CVT_DEVNAM : CVT_DEVNAM ADDRESSING_MODE(ABSOLUTE) ;



!
! **********************************************************************
! **********************************************************************
! TEMPORARY CLUGE UNTIL THESE DEFINITIONS CAN BE PUT INTO STARLET/SYSDEF
! **********************************************************************
! **********************************************************************
LITERAL CJF_EVENT_FLAG = 25 ;

        LINKAGE
            LINKALOP1IMAG = JSB ( REGISTER = 1 ; REGISTER = 1, REGISTER = 2 )
                            : NOPRESERVE ( 3 ) ;

        EXTERNAL ROUTINE
            EXE$ALOP1IMAG : LINKALOP1IMAG ;

        LINKAGE
            LINKDEAP1 = JSB ( REGISTER = 0 , REGISTER = 1 )
                            : NOPRESERVE ( 0, 1, 2, 3 ) ;

        EXTERNAL ROUTINE
            EXE$DEAP1 : LINKDEAP1 ;
```

```
MACRO
    DO_BINDS ( BASE_ADDR, NAME, TYPE, LENGTH ) [] =
        BIND NAME = BASE_ADDR + DOBIND_OFFSET : %REMOVE (TYPE) ;
        %ASSIGN ( DOBIND_OFFSET, DOBIND_OFFSET + LENGTH )
        DO_BINDS ( BASE_ADDR, %REMAINING ) %;

MACRO
    DEFINE_OFFSETS ( BASE_ADDR ) =
        %IF NOT %DECLARED(DOBIND_OFFSET)
        %THEN
        COMPILETIME
            DOBIND_OFFSET = 0 ;
        %FI
        DO_BINDS ( BASE_ADDR, %REMAINING ) %;

!+
! ALLOCATE_P1 - Allocate memory from P1
!               DTA_LNGTH       = number of bytes to allocate
!               ADDR_SIZ_BLCK   = address of a two longword block
!                                 to recieve the address and size
!                                 of the allocated memory.
!-
MACRO
    ALLOCATE_P1 ( DTA_LNGTH, ADDR_SIZ_BLCK ) =
        BEGIN
        !+
        ! Get a block of memory to hold our data.  Make sure that
        ! there is enough for the length specified plus the staging data.
        !-
        MAP ADDR_SIZ_BLCK : VECTOR [,LONG] ;

        BIND
            ALLOC_SIZ  = ADDR_SIZ_BLCK [1] : LONG,
            ALLOC_ADDR = ADDR_SIZ_BLCK [0] : LONG ;

        LOCAL
            RET_STAT : LONG ;

        RET_STAT = EXE$ALOP1IMAG ( DTA_LNGTH ; ALLOC_SIZ, ALLOC_ADDR );
        IF .RET_STAT
            THEN CH$FILL ( 0, .ALLOC_SIZ, .ALLOC_ADDR ) ;

        .RET_STAT

        END %;

!+
! DEALLOCATE_P1 - Deallocate memory from P1
!                 ADDR_SIZ_BLCK   = address of a two longword block
!                                   which contains the address and size
!                                   of the allocated memory.
!-
MACRO
    DEALLOCATE_P1 ( ADDR_SIZ_BLCK ) =
```

```
        BEGIN
        BIND DATA_BLCK = ADDR_SIZ_BLCK : VECTOR [,LONG] ;

        EXE$DEAP1 ( .DATA_BLCK[0], .DATA_BLCK[1] )

        END %;
```

```
!++
! INIT_STAGE - Initialize staging
!
!       The INIT_STAGE macro establishes the calling routine as a
!       recoverable entity.  Its only argument is the name of a routine
!       (not called, just for show) to undo the entire action of the
!       routine to follow the INIT_STAGE call.  The INIT_STAGE macro MUST
!       be positioned precisely between the blocks declarations, and
!       executables.
!
!       If an error is signalled, the appropriate code (as specified
!       in the most recent NEXT_STAGE macro in the current
!       routine) is executed.  Useage of this macro must be non-reentrant.
!
!       Any data which is required by the recovery code must either be global,
!       or specified in the SAVE_DATA parameter list passed to the NEXT_STAGE
!       macro.  Each item in the data vector is stored as a longword.
!
!       The code to be executed is specified in the CODE parameter of the
!       NEXT_STAGE macro.  In places where the code must reference the
!       data from the DATA vector, the format is: .DATA[loc-in-vector].
!       "loc-in-vector" is the parameter number within the DATA vector
!       relative to zero.
!
!       If appropriate, the call to NEXT_STAGE may include PREVIOUS=when
!       where "when" can be:
!               BEFORE  execute the previous NEXT_STAGE code BEFORE executing
!                       code from this call
!               AFTER   execute the previous NEXT_STAGE code AFTER executing
!                       code from this call
!               NEVER   do not (NEVER) execute the previous NEXT_STAGE code
!                       this is the default.
!               FINAL   remove the previously performed NEXT_STAGE macro
!--

MACRO
    INIT_STAGE ( A1, A2, A3, A4, A5, A6, A7, A8, A9 ) =
        EXTERNAL ROUTINE
            COND_HANDLER ,
            GET_PC ;

        OWN
            ENAB_V_STAGE_LIST   : VOLATILE LONG ,
            STAGE_BLK           : VOLATILE VECTOR [3, LONG],
            STAGE_DATA_AREA     : VOLATILE VECTOR [CJF$C_MAX_DATA_AREA, BYTE],
            STAGE_DATA_OFFSET   : VOLATILE LONG,
            STAGE_LIST_PTR      : VOLATILE LONG,
            STAGE_LIST          : VOLATILE VECTOR[ CJF$C_MAX_STAGE*12, BYTE ] ;

        ENABLE COND_HANDLER (ENAB_V_STAGE_LIST,STAGE_LIST_PTR) ;

        BUILTIN
            FP ;
```

```
        %IF NOT %DECLARED(UNIQUE_NUMBER)
        %THEN
            COMPILETIME UNIQUE_NUMBER = 0 ;
        %FI

        %IF NOT %DECLARED(LAST_STAGE_NUMBER)
        %THEN
            COMPILETIME LAST_STAGE_NUMBER = 0 ;
        %FI

        STAGE_DATA_OFFSET = 0 ;

        %IF JNLACP_BUILD
        %THEN
        STAGE_BLK[0] = STAGE_LIST_PTR ;
        STAGE_BLK[1] = STAGE_LIST ;
        STAGE_BLK[2] = .GL_STAGEBLK ;
        GL_STAGEBLK  = STAGE_BLK;
        %FI

        CH$FILL ( 0, (CJF$C_MAX_STAGE * 12), STAGE_LIST ) ;
        ENAB_V_STAGE_LIST = STAGE_LIST ;
        STAGE_LIST_PTR    = STAGE_LIST ;
    % ;
```

```
!++
! SERVICE_INIT_STAGE
!
!       This macro when coupled with the DEFI_STAGE_DATA macro and the
! SERVICE_END_STAGE macro perform the same functions as the INIT_STAGE
! macro but they may be used in the reentrant service code.  The service
! must first call SERVICE_INIT_STAGE to enable the condition handler and
! perform an expand region to get data space in P0.  A BEGIN block must then
! be started and DEFI_STAGE_DATA must be called before the first executable
! statement within the block but after the last declaration.  Finally, just
! before the service is done it must call SERVICE_END_STAGE to delete the
! newly aquired P0 space.
!
!       It is also highly recommended that a NEXT_STAGE routine is declared
! after DEFI_STAGE_DATA to delete the virtual address space aquired by
! SERVICE_INIT_STAGE.  If such a NEXT_STAGE is used it MUST be the very
! last next stage routine to be executed by the condition handler otherwise
! the condition handler itself will access violate.  This is due to the fact
! that the memory locations referenced by the condition handler are in the
! newly aquired address space so if you delete the address space before the
! condition handler is done then all hell will break loose.
!
! Suggested form of the NEXT_STAGE declaration :
!
!       NEXT_STAGE ( SAVE_DATA = ( ADDR_SIZ_BLCK ),
!                    CODE      = ( SERVICE_END_STAGE ( .DATA[0] )),
!                    PREVIOUS  = NEVER ) ;
!--
MACRO
    SERVICE_INIT_STAGE ( DTA_LNGTH, ADDR_SIZ_BLCK, ALLOC_STAT ) =
        EXTERNAL ROUTINE
            COND_HANDLER ,
            GET_PC ;

        BUILTIN
            FP ;

        LOCAL
            ENAB_V_STAGE_LIST   : VOLATILE LONG ,
            STAGE_DATA_OFFSET   : VOLATILE LONG ,
            STAGE_LIST_PTR      : VOLATILE LONG ;

        LITERAL
            ROUNDED_SIZE = (((CJF$C_MAX_STAGE*12) + 12 +
                             CJF$C_MAX_DATA_AREA + DTA_LNGTH) + 7 )
                           AND %X'FFFFFFF8' ;

        ENABLE COND_HANDLER (ENAB_V_STAGE_LIST,STAGE_LIST_PTR) ;

        %IF NOT %DECLARED(UNIQUE_NUMBER)
        %THEN
            COMPILETIME UNIQUE_NUMBER = 0 ;
        %FI

        %IF NOT %DECLARED(LAST_STAGE_NUMBER)
        %THEN
```

```
        COMPILETIME LAST_STAGE_NUMBER = 0 ;
    %FI


    %IF %DECLARED ( DOBIND_OFFSET )
    %THEN %ASSIGN ( DOBIND_OFFSET, 0 )
    %FI

    COMPILETIME
        SERV_INIT_DONE = 0 ;

    STAGE_DATA_OFFSET = 0 ;

    !+
    ! Get a block of memory to hold our data.  Make sure that
    ! there is enough for the length specified plus the staging data.
    !-
    ALLOC_STAT = ALLOCATE_P1 ( ROUNDED_SIZE, ADDR_SIZ_BLCK ) ;
    IF NOT .ALLOC_STAT
        THEN ERR_EXIT ( SS$_INSFMEM ) %;


MACRO
    DEFI_STAGE_DATA ( BASE_ADR, A2, A3, A4, A5, A6, A7, A8, A9 ) =

    %IF NOT %DECLARED (SERV_INIT_DONE)
    %THEN %EXITMACRO
    %FI
    !+
    ! ********************** CAUTION **********************
    ! If the size of the required data storage is changed
    ! it must also be reflected in the SERVICE_INIT_STAGE
    ! macro.
    ! ********************** CAUTION **********************
    !-
    DEFINE_OFFSETS (BASE_ADR,
        STAGE_BLK           , (VOLATILE VECTOR[,LONG]), 12,
        STAGE_DATA_AREA     , (VOLATILE VECTOR[,BYTE]),CJF$C_MAX_DATA_AREA,
        STAGE_LIST          , (VOLATILE VECTOR[,BYTE]),CJF$C_MAX_STAGE*12);

    %IF JNLACP_BUILD
    %THEN
    STAGE_BLK[0] = STAGE_LIST_PTR ;
    STAGE_BLK[1] = STAGE_LIST ;
    STAGE_BLK[2] = .GL_STAGEBLK ;
    GL_STAGEBLK  = STAGE_BLK;
    %FI

    CH$FILL ( 0, (CJF$C_MAX_STAGE * 12), STAGE_LIST ) ;
    ENAB_V_STAGE_LIST = STAGE_LIST ;
    STAGE_LIST_PTR    = STAGE_LIST ;
%;

MACRO
    SERVICE_END_STAGE ( ADDR_SIZ_BLCK ) =
        DEALLOCATE_P1 ( ADDR_SIZ_BLCK )%;
```

```
!++
! NEXT_STAGE - Declare next stage and recovery data and code
!
!--
!
KEYWORDMACRO
    NEXT_STAGE ( SAVE_DATA, CODE, PREVIOUS=NEVER )  =

        ! Check for previous stage removal request
        !
        %IF %IDENTICAL ( PREVIOUS, FINAL )
        %THEN
            %IF NOT %NULL(SAVE_DATA) OR NOT %NULL (CODE)
            %THEN %ERRORMACRO ('Cannot add staging while removing previous stage') ;
            %FI ;

            %PRINT ('------------------------------------>> STAGE_',
                %NUMBER(LAST_STAGE_NUMBER),'  REMOVED '
                ' <<------------------------------------')

            %ASSIGN (LAST_STAGE_NUMBER,LAST_STAGE_NUMBER-1)

            STAGE_LIST_PTR = .STAGE_LIST_PTR - 12 ;

            IF .STAGE_LIST_PTR LSSU STAGE_LIST
            THEN STAGE_LIST_PTR = STAGE_LIST ;

            %EXITMACRO ;
        %FI

            BEGIN
            SWITCHES LIST(NOOBJECT);

            ! Create unique number to label storage locations
            !
            %IF DEBUG_PREFIX_COMPILE
            %THEN
                %PRINT(' Unique: ',%NUMBER(UNIQUE_NUMBER),' before increment')
            %FI
            %ASSIGN (UNIQUE_NUMBER,UNIQUE_NUMBER+1)
            %ASSIGN (LAST_STAGE_NUMBER,UNIQUE_NUMBER)
            %PRINT ('------------------------------------>> STAGE_',
                %NUMBER(UNIQUE_NUMBER),
                ' <<------------------------------------')

            ! If there is code, put it in a routine.
            ! If not, define routine address as 0
            !
            %IF NOT %NULL( CODE )
            %THEN
                ROUTINE %NAME('STAGE_',%NUMBER(UNIQUE_NUMBER))
                    (DATA_LOC, SIG_V : REF VECTOR, MECH_V : REF VECTOR,
                    ENAB_V : REF VECTOR ) : NOVALUE =

                        BEGIN
                        BIND
```

```
                    DATA = .DATA_LOC    : VECTOR [,LONG];
                LINES_OF_CODE (%REMOVE CODE )  ;
                END  ;
%ELSE
    BIND %NAME('STAGE_',%NUMBER(UNIQUE_NUMBER)) = 0 ;
%FI

! If we will NEVER call the previously registered
! recovery routine, reset subroutine stack
!
%IF %IDENTICAL ( PREVIOUS, NEVER )
%THEN
    STAGE_LIST_PTR = STAGE_LIST ;
%FI

! Create the data vector and fill it.  Put data address in
! subroutine stack.
!
DATA_VECTOR ( SAVE_DATA )  ;

! Put routine info in the subroutine stack
!
.STAGE_LIST_PTR + 0 = %NAME('STAGE_',%NUMBER(UNIQUE_NUMBER)) ;
.STAGE_LIST_PTR + 8 =
    %IF %IDENTICAL(PREVIOUS,BEFORE)
    %THEN
    -1
    %ELSE %IF %IDENTICAL(PREVIOUS,AFTER)
    %THEN
    +1
    %ELSE %IF %IDENTICAL(PREVIOUS,NEVER)
    %THEN
    0
    %ELSE 0 %ERROR ('Illegal previous indicator on STAGE_',
        %NUMBER(UNIQUE_NUMBER) )
    %FI %FI %FI ;

! Bump subroutine stack pointer and check for errors
!
STAGE_LIST_PTR = .STAGE_LIST_PTR + 12 ;
%IF UNIQUE_NUMBER GTR CJF$C_MAX_STAGE
%THEN %ERROR ('CJF$C_MAX_STAGE is too low')
%FI
IF (.STAGE_LIST_PTR-STAGE_LIST)/12 GTR CJF$C_MAX_STAGE
THEN
    ! This maybe should be some other error
    !
    ERR_EXIT ( CJF$_OVERSTAGE ) ;

END ;
% ;

MACRO
    LINES_OF_CODE [ LINE_OF_CODE ] =
            LINE_OF_CODE
    %,
```

```
    DATA_VECTOR ( DATA_ITEMS ) =
        %IF %NULL(DATA_ITEMS)
        %THEN
            %EXITMACRO
        %FI

        BEGIN
        BIND
            %NAME ( 'STAGE$DATA_', %NUMBER(UNIQUE_NUMBER) ) =
                STAGE_DATA_AREA + .STAGE_DATA_OFFSET
                    : VECTOR [ NUM_PARAMS ( %REMOVE DATA_ITEMS ),LONG ] ;

        COMPILETIME
            VECTOR_LOC = 0 ;

        STAGE_DATA_OFFSET =
                NUM_PARAMS ( %REMOVE DATA_ITEMS )*4 + .STAGE_DATA_OFFSET ;

        IF .STAGE_DATA_OFFSET GTR CJFSC_MAX_DATA_AREA
        THEN ERR_EXIT(CJFS_INTERNAL);      ! ('CJFSC_MAX_DATA_AREA is too low')

        .STAGE_LIST_PTR + 4 = %NAME ( 'STAGE$DATA_', %NUMBER(UNIQUE_NUMBER) ) ;

        DATA_FILL ( %REMOVE DATA_ITEMS )  ;
        END ;
    % .

    DATA_FILL [ DATEM ] =
        %NAME ( 'STAGE$DATA_', %NUMBER(UNIQUE_NUMBER) ) [ VECTOR_LOC ] = DATEM
        %ASSIGN ( VECTOR_LOC, VECTOR_LOC + 1 )
    % .

    NUM_PARAMS ( ITEMS ) =
        %LENGTH
    % ;
```

```
!++
! ADD_PRIV      - save current privs and add specified ones to process
! RESTORE_PRIV  - restore previous privs
!
!       The user should assure that there be no possible code path
!       that executes an ADD_PRIV without THE MATCHING
!       RESTORE_PRIV.
!
!--

MACRO
    ADD_PRIV ( FIRST_PARAM ) =
        BEGIN

        LOCAL
            NEW_PRIVS   : BBLOCK [8],
            OFF_PRIVS   : BBLOCK [8],
            PREV_PRIVS  : BBLOCK [8] ;

        BIND
            NEW_BITS    = NEW_PRIVS     : BITVECTOR ,
            OFF_BITS    = OFF_PRIVS     : BITVECTOR ,
            PREV_BITS   = PREV_PRIVS    : BITVECTOR ;

        CH$FILL ( 0 , 8 , NEW_PRIVS ) ;

        MAKE_PRIV_MASK( FIRST_PARAM, %REMAINING ) ;

        $SETPRV ( ENBFLG= 1, PRVADR= NEW_PRIVS, PRMFLG= 0, PRVPRV= PREV_PRIVS ) ;
    % ,

    MAKE_PRIV_MASK [ PRIV ] =

        NEW_PRIVS [ %NAME ( 'PRV$V_', PRIV ) ] = 1 ;
% ,

RESTORE_PRIV =
    INCR PRIV_NUM FROM 0 TO 63 BY 1 DO
        OFF_BITS [ .PRIV_NUM ] =
            .NEW_BITS [ .PRIV_NUM ] AND NOT .PREV_BITS [ .PRIV_NUM ] ;

    $SETPRV ( ENBFLG=0, PRVADR= OFF_PRIVS, PRMFLG= 0 ) ;
    END ;
% ;
```

```
!++
! TEST_PRIV - Test if user has the priv
!
!--

MACRO
    TEST_PRIV ( PRIV ) =
        BEGIN
        LOCAL
            CUR_PRIVS    : BBLOCK [8] ;
        $SETPRV ( PRVPRV= CUR_PRIVS ) ;
        .CUR_PRIVS [ %NAME ( 'PRV$V_',PRIV ) ]
        END
    % ;
```

```
!++
! ACCESS_ALLOWED - probe memory location
!
! parameters are:
!
!       BASE    base address
!       LENGTH  length of region
!       RW      either R or W for Read or Write
!--
!
MACRO
    ACCESS_ALLOWED ( BASE, LENGTH, RW ) =

        %IF (NOT %IDENTICAL(RW,R)) AND (NOT %IDENTICAL(RW,W))
        %THEN
            %ERROR ('RW (thrird) parameter must be either R or W')
            0 ;
            %EXITMACRO
        %FI

        BEGIN

        LOCAL
            PSL : BBLOCK [ 4 ] ,
            MODE ;

        BUILTIN
            PROBEW ,
            PROBER ,
            MOVPSL ;

        MOVPSL (PSL) ;

        MODE = .PSL [ PSL$V_PRVMOD ] ;

        %NAME ('PROBE',RW) ( MODE, %REF(LENGTH), BASE )

        END  % ;
```

```
!++
! ARG_CHECK - Validate number of arguments
!
!       Macro to validate that correct number of arguments were specified
!       on a routine call
!
!--

MACRO   ARG_CHECK ( NUM ) =
    BEGIN
    BUILTIN ACTUALCOUNT ;

    IF ACTUALCOUNT() LSS NUM
        THEN ERR_EXIT ( SS$_INSFARG ) ;
    IF ACTUALCOUNT() GTR NUM
        THEN ERR_EXIT ( CJFS_OVRMAXARG ) ;

    END ; %;
```

```
!++
! KERNEL_CALL - Call kernel mode routine
!
! Macro to call the change mode to kernel system service.
! Macro call format is "KERNEL_CALL (ROUTINE, ARG1, ARG2, ... )".
!
! ***** Note: The following macro violates the Bliss language definition
! ***** in that it maks use of the value of SP while building the arg list.
! ***** It is the opinion of the Bliss maintainers that this usage is safe
! ***** from planned future optimizations.
!
!--

MACRO
        KERNEL_CALL (R) =
            BEGIN

            EXTERNAL ROUTINE
                SYS$CMKRNL ;

            EXTERNAL ROUTINE
                CMOD$SETEXV ;

            BUILTIN SP;

            SYS$CMKRNL ( CMOD$SETEXV, .SP, %LENGTH+1, R, .SP, %LENGTH-1
                %IF %LENGTH GTR 1 %THEN ,%REMAINING %FI)

            END  %;
    END  %;
```

```
!++
! SET_IPL - Set processor priority
!
!--
!
MACRO
        SET_IPL (LEVEL) =
                BEGIN
                BUILTIN MTPR;
                MTPR (%REF (LEVEL), PR$_IPL)
                END
                %;
```

```
!++
! SOFT_INT - force software interrupt
!--

MACRO
        SOFT_INT (LEVEL) =
                MTPR(%REF(LEVEL),PR$_SIRR)%;
```

```
!++
! ERR_EXIT - Error exit macro.
!--

! This definition could be used for Journal ACP, since no user CHMU handler
! could be there yet

MACRO
    ERR_EXIT (CODE) =
        BEGIN

        GLOBAL REGISTER
            R0  = 0 ;

        %IF NOT %NULL (CODE)
        %THEN
            R0 = CODE ;
        %ELSE
            R0 = 0 ;
        %FI

         CHMU ( %REF ( 0 )  ) ;

         %IF NOT %NULL (%REMAINING)
         %THEN %WARNING ('Additional arguments not allowed on this call')
         %FI
        END

        %,

! This defintion is used for both services and Journal ACP.

MACRO   ERR_EXIT(CODE) =
        SIGNAL(%IF NOT %NULL(CODE)
                        %THEN CODE %ELSE 0 %FI
                        %IF NOT %NULL(%REMAINING)
                        %THEN ,%REMAINING %FI)
                %,

        ERR_MESSAGE [] =
                SIGNAL (%REMAINING)
        %;
```

```
!++
! BUG_CHECK - Macro used to signal fatal errors (internal consistency checks).
!--

MACRO
        BUG_CHECK (CODE, TYPE, MESSAGE) =
                BEGIN
                BUILTIN BUGW;
                EXTERNAL LITERAL %NAME('BUG$_',CODE);
                BUGW (%NAME('BUG$_',CODE) OR 4)
                END
                %;

!++
!
! CJF$UNLOCK_PROTO
!
!       This macro ends the synchronization on the proto UCB by
!       dequeing the specified lock.
!
!********************* W A R N I N G ********************************
!
! This macro is duplicated in [JCP.SRC]JCPREQ.R32 .  If any changes
! are made to the macro make sure that they are also reflected in
! the JCP's require file.
!
!********************* W A R N I N G ********************************
!--
KEYWORDMACRO CJF$UNLOCK_PROTO ( LOCK_ID ) =
    $DEQ ( LKID = .LOCK_ID )% ;
```

```
!+
! Define file block fields
!-
FIELD
    !+
    ! ********** CAUTION ***********
    ! IOCHAN must be the first field
    ! ********** CAUTION ***********
    !-
    FILE_BLCK_FIELDS =
        SET
        IOCHAN  = [0,0,16,0],
        DIR_FID = [2,0,0,0],
        CREDAT  = [8,0,0,0],
        FIB     = [16,0,0,0],
        RECATTR = [16+FIB$C_LENGTH,0,0,0]
        TES;
LITERAL FILBLK_ENTLEN = FIB$C_LENGTH + FAT$C_LENGTH + 16 ;
STRUCTURE
        FILEBLOCK [I, O, P, S, E; N] =
                [N * FILBLK_ENTLEN]
                (FILEBLOCK + (I * FILBLK_ENTLEN) + O)<P,S,E>;


!++
! Defines UIC group and member fields.
!--
FIELD
    UIC_FIELDS =
        SET
        MEMBER = [0,0,16,0],
        GROUP  = [0,16,16,0]
        TES;

!++
! This defines a DESCRIPTOR data structure
!
!--
FIELD
    DESCR_FIELDS =                          ! Define the fields for a DESCRIPTOR
        SET
        LENGTH  = [0, 0, 16, 0],
        DTYPE   = [0, 16, 8, 0],
        CLASS   = [0, 24, 8, 0],
        POINTER = [1, 0, 32, 0]
        TES;

MACRO
    CDESCRIPTOR = BLOCK[2] FIELD(DESCR_FIELDS)%;

FIELD
    CDDESCR_FIELDS =
        SET
        OFFSET    = [0, 0, 16, 0],
        SIZE      = [0, 16, 16, 0],
        USER_ADDR = [1, 0, 32, 0]
```

```
        TES;

MACRO
    CDDESCRIPTOR = BLOCK[2] FIELD(CDDESCR_FIELDS)%;

!
! Macro to generate a string with a descriptor.
!
MACRO
        DESCRIPTOR (STRING) =
                UPLIT (%CHARCOUNT (STRING), UPLIT BYTE (STRING))%;
```

```
!++
! Structure for all MDL defined blocks.
!
!--
!
STRUCTURE
        BBLOCK [O, P, S, E; N] =
                [N]
                (BBLOCK+O)<P,S,E>,

        BBLOCKVECTOR [I, O, P, S, E; N, BS] =
                [N*BS]
                ((BBLOCKVECTOR+I*BS)+O)<P,S,E>;
```

```
!++
! global literals
!
!--
!
LITERAL
    JNL$C_MAX_COPIES = 1      ,     ! max # of jnl copies
    JNL$C_MAX_FILLEN = 255 ,        ! maximum filename string length
    JNL$C_MAX_BUFSIZ = 5      ,     ! maximum # of 512 byte blocks per buffer
    JNL$C_MAX_MAXSIZ = 32767 ,      ! maximum record size
    JNL$C_DEFBSIZ    = 512 ,        ! default I/O buffer size (in bytes)
    JNL$C_MAX_JNLS   = 30 ,         ! maximum number of journals on one tape
    NAMTBL_BUFF_BLKS = MAX ( 2,
                ((NTE$C_MAXREC + NTE$C_BLKSIZ - 1) / NTE$C_BLKSIZ) + 1)),
    NAMTBL_BUFF_LEN  = NAMTBL_BUFF_BLKS * NTE$C_BLKSIZ ;
```

JNLBUSR
R32

JNLDEFINT
SDL

CJFV4_

JNLPREFIX
R32

CJFRUFMAC
SDL

RUFUSR
SDL

JNLFILE
SDL

UPGRADE
LIS

BOPTIONS
R32

JNLDEF
SDL

BACKUP CLD
DCLTABLES CLD
DISMOUNT CLD
ENCRYPT CLD
REPLY CLD
CLD
CREATE CLD
MCRINT CLD
SET CLD
DCLTABLES MAP
DEF CLD
DMO CLD
LIBRARIAN CLD
PASCAL CLD
RUN CLD
ACC CLD
BAD CLD
DCLINT CLD
DELETE CLD
DUMP CLD
EXCHANGE CLD
MCRTABLES CLD
PATCH CLD
CHECKSUM CLD
ANALYZE CLD
EDIT CLD
FORTRAN CLD
LINK CLD
MESSAGE CLD
PHONE CLD
CLISYM1 CLD
MCRSET CLD
RUNOFF CLD
PSECTS R32
DIFF CLD
MONITOR CLD
RECOVER CLD
MCRTABLES MAP
CONVERT CLD
HELP CLD
MACRO CLD
DIRECTORY CLD
SDL CLD
JNLUSR MAR
COPY CLD
EDT CLD
INIT CLD
MOUNT CLD
RENAME CLD
MAIL CLD
SEARCH CLD